```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE GENCMD
OBJECT MODULE PLACED IN GENCMD.OBJ
COMPILER INVOKED BY:  :F0: GENCMD.PLM DATE( MARCH 16, 1982 ) XREF OPTIMIZE(3) DEBUG


  1          GENCMD:
             DO;
             /*   CP/M 8086 CMD file generator

                  COPYRIGHT (C) 1981
                  DIGITAL RESEARCH
                  BOX 579 PACIFIC GROVE
                  CALIFORNIA 93950

                  */

  2   1      DECLARE
                  digital$code literally '0081h',  /*DR code record */
                  digital$data literally '0082h',  /* DR data record */
                  digital02 literally '0085h',   /* DR 02 records */
                  paragraph literally '16',
                  ex literally '12',         /* extent */
                  nr literally '32',         /* current record */
                  maxb address external,
                  fcba(33) byte external,    /* DEFAULT FILE CONTROL BLOCK */
                  buffa(128) byte external; /* DEFAULT BUFFER ADDRESS */


  3   1      DECLARE COPYRIGHT(*) BYTE DATA
                  (' COPYRIGHT (C) 1981, DIGITAL RESEARCH ');

  4   1      MON1: PROCEDURE(F,A) EXTERNAL;
  5   2          DECLARE F BYTE, A ADDRESS;
  6   2          END MON1;

  7   1      MON2: PROCEDURE(F,A) BYTE EXTERNAL;
  8   2          DECLARE F BYTE, A ADDRESS;
  9   2          END MON2;

 10   1      DECLARE SP ADDRESS;

 11   1      BOOT: PROCEDURE;
 12   2          call mon1 (0,0);
 13   2          END BOOT;

 14   1      declare segmts(11) structure (name(5) byte,begin$add address)
                  initial ('CODE ',00h,'DATA ',0ffffh,'EXTRA',0ffffh,'STACK',0,
                    'X1   ',0,'X2   ',0,'X3   ',0,'X4   ',0,'8080 ',0,'NZERO',0,
                    'NHEAD',0);


 15   1      declare header (15) structure
                (typseg byte,file$length address,absolute$add address,
                  minimum$mem address,
                  maximum$mem address) initial (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

CP/M-86 v.1.1 (Vol. II)
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

```
            0,0,0,
```

```
 49  2     CRLF: PROCEDURE;
 50  3         CALL PRINTCHAR(CR);
 51  3         CALL PRINTCHAR(LF);
 52  3         END CRLF;

 53  2     PRINTNIB: PROCEDURE(N);
 54  3         DECLARE N BYTE;
 55  3             IF N > 9 THEN CALL PRINTCHAR(N+'A'-10); ELSE
 57  3             CALL PRINTCHAR(N+'0');
 58  3         END PRINTNIB;

 59  2     PRINTHEX: PROCEDURE(B);
 60  3         DECLARE B BYTE;
 61  3             CALL PRINTNIB(SHR(B,4)); CALL PRINTNIB(B AND 0FH);
 63  3         END PRINTHEX;

 64  2     PRINTADDR: PROCEDURE(A);
 65  3         DECLARE A ADDRESS;
 66  3         CALL PRINTHEX(HIGH(A)); CALL PRINTHEX(LOW(A));
 68  3         END PRINTADDR;

 69  2     PRINTM: PROCEDURE(A);
 70  3         DECLARE A ADDRESS;
 71  3         CALL MON1(9,A);
 72  3         END PRINTM;

 73  2     PRINT: PROCEDURE(A);
 74  3         DECLARE A ADDRESS;
             /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
             NEXT DOLLAR SIGN IS ENCOUNTERED WITH PRECEDING CRLF */
 75  3         CALL CRLF;
 76  3         CALL PRINTM(A);
 77  3         END PRINT;

 78  2     declare    abuffadr address,
                LA ADDRESS;  /* CURRENT LOAD ADDRESS */
 79  2          declare head byte;

 80  2     PERROR: PROCEDURE(A);
             /* PRINT ERROR MESSAGE */
 81  3         DECLARE A ADDRESS;
 82  3         CALL PRINT(.('ERROR: $'));
 83  3         CALL PRINTM(A);
 84  3         CALL PRINTM(.(', LOAD ADDRESS $'));
 85  3         CALL PRINTADDR(LA);
 86  3         CALL BOOT;
 87  3         END PERROR;


 88  2     diskerror: procedure;
 89  3              call perror(.('DISK WRITE$'));
 90  3         end diskerror;

 91  2     DECLARE DCNT BYTE;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 92  2     setdma: procedure(a);
 93  3         declare a address;
 94  3         call mon1 (26,a);
 95  3         end setdma;

 96  2     OPEN: PROCEDURE(FCB);
 97  3         DECLARE FCB ADDRESS;
 98  3         DCNT = MON2(15,FCB);
 99  3         END OPEN;

100  2     CLOSE: PROCEDURE(FCB);
101  3         DECLARE FCB ADDRESS;
102  3         DCNT = MON2(16,FCB);
103  3         END CLOSE;

104  2     SEARCH: PROCEDURE(FCB);
105  3         DECLARE FCB ADDRESS;
106  3         DCNT = MON2(17,FCB);
107  3         END SEARCH;

108  2     SEARCHN: PROCEDURE;
109  3         DCNT = MON2(18,0);
110  3         END SEARCHN;

111  2     DELETE: PROCEDURE(FCB);
112  3         DECLARE FCB ADDRESS;
113  3         CALL MON1(19,FCB);
114  3         END DELETE;

115  2     DISKREAD: PROCEDURE(FCB) BYTE;
116  3         DECLARE FCB ADDRESS;
117  3         RETURN MON2(20,FCB);
118  3         END DISKREAD;

119  2     DISKWRITE: PROCEDURE(FCB) BYTE;
120  3         DECLARE FCB ADDRESS;
121  3         RETURN MON2(21,FCB);
122  3         END DISKWRITE;

123  2     MAKE: PROCEDURE(FCB);
124  3         DECLARE FCB ADDRESS;
125  3         DCNT = MON2(22,FCB);
126  3         END MAKE;

127  2     RENAME: PROCEDURE(FCB);
128  3         DECLARE FCB ADDRESS;
129  3         CALL MON1(23,FCB);
130  3         END RENAME;

131  2         MOVE: PROCEDURE(S,D,N);
132  3             DECLARE (S,D) ADDRESS, N BYTE,
                  A BASED S BYTE, B BASED D BYTE;
133  3                 DO WHILE (N:=N-1) <> 255;
134  4                 B = A; S=S+1; D=D+1;
137  4                 END;
138  3             END MOVE;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
139  2        declare char byte;



140  2    comline$error: procedure;
141  3        declare i byte;
142  3        call crlf;
143  3        do i = 1 to tbp;
144  4           call printchar (buffer(i));
145  4           end;
146  3        call printchar ('?');
147  3        call crlf;
148  3        call boot;
149  3        end comline$error;



150  2    retchar: procedure byte;
            /* get another character from command tail */
151  3           if (tbp :=tbp+1) <= count$command$tail then
152  3              return buffer(tbp);
153  3           else return (0dh);
154  3           end retchar;

155  2    tran: procedure(b) byte;
156  3           declare b byte;
157  3           if b < ' ' then return 0dh;  /* non-graphic */
159  3           if b - 'a' < ('z' - 'a') then
160  3              b = b and 101$1111b;    /* upper case */
161  3           return b;
162  3           end tran;

163  2    next$non$blank: procedure;
164  3           char=tran(retchar);
165  3           do while char= ' ';
166  4           char= tran(retchar);
167  4           end;
168  3        end next$non$blank;

169  2        CHECK$ONE$HEX: PROCEDURE (h) BYTE;
                /* READ ONE HEX CHARACTER FROM THE INPUT */
170  3           DECLARE H BYTE;
171  3           IF H  - '0' <= 9 THEN RETURN H - '0';
173  3           IF H - 'A' > 5 THEN
174  3           return (0ffh);
175  3           RETURN H - 'A' + 10;
176  3           END CHECK$ONE$HEX;



177  2        MAKE$DOUBLE: PROCEDURE(H,L) ADDRESS;
                /* CREATE A DOUBLE BYTE VALUE FROM TWO SINGLE BYTES */
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
178  3          DECLARE (H,L) BYTE;
179  3          RETURN SHL(DOUBLE(H),8) OR L;
180  3          END MAKE$DOUBLE;



181  2     delimiter: procedure byte;    /*  logical    */
182  3        declare i byte;
183  3        declare del (*) byte data (0dh,'[], ');
184  3        do i = 0 to last(del);
185  4           if char = del(i) then return true;
187  4        end;
188  3        return false;
189  3        end delimiter;



190  2     get$num: procedure address;
191  3        declare paradd address;
192  3        paradd = 0;
193  3        char = retchar;
194  3        do while not delimiter ;
195  4           if (char:=check$one$hex(char)) = 0ffh then
196  4           call comline$error;   else
197  4           paradd = paradd * 16 + char;
198  4           char = retchar;
199  4           end;

200  3         return paradd;
201  3        end get$num;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
202  2        GETCHAR: PROCEDURE BYTE;
                 /* GET NEXT CHARACTER FROM DISK BUFFER */
203  3           DECLARE I BYTE;
204  3           IF (SBP := SBP+1) <= LAST(SBUFF) THEN
205  3              RETURN SBUFF(SBP);
                 /* OTHERWISE READ ANOTHER BUFFER FULL */
206  3              DO SBP = 0 TO LAST(SBUFF) BY 128;
207  4              IF (I:=DISKREAD(.SFCB)) = 0 THEN
208  4                 CALL MOVE(.buffer,.SBUFF(SBP),80H); ELSE
209  4                 DO;
210  5                 IF I<>1 THEN CALL PERROR(.('DISK READ$'));
212  5                 SBUFF(SBP) = EOFILE;
213  5                 SBP = LAST(SBUFF);
214  5                 END;
215  4              END;
216  3           SBP = 0; RETURN SBUFF(0);
218  3           END GETCHAR;
219  2     DECLARE
           STACKPOINTER LITERALLY 'STACKPTR';

          /* INTEL HEX FORMAT LOADER */

220  2     RELOC: PROCEDURE;
221  3        DECLARE (RL, CS, RT,K) BYTE;
```

```
222  3        declare multi$segments byte;
223  3        DECLARE
                 tabs address,   /* temporary value */
                 TA ADDRESS,     /* TEMP ADDRESS */
                 SA ADDRESS,     /* START ADDRESS */
                 FA ADDRESS,     /* FINAL ADDRESS */
                 NB ADDRESS,     /* NUMBER OF BYTES LOADED */
                 nxb byte,       /*  next byte in stream   */
                 segadjst address   /* segment adjust  */
                 seg$length (8) address,    /* length of each segment  */
                 write$add address,

                 MBUFF based mbuffadr (256) BYTE,
                 P BYTE;
224  3        declare high$add address;

225  3        SETMEM: PROCEDURE(B);
                /* set mbuff to b at location la */
226  4           DECLARE (B) BYTE;
227  4           if (.memory+la) > maxb then
228  4           do;
229  5             call print (.('INSUFFICIENT MEMORY TO CREATE CMD FILE $'));
230  5             call boot;
231  5             end;
232  4           MBUFF(LA) = B;
233  4           END SETMEM;


234  3        zero$mem: procedure;
235  4           do while (.memory +la) <maxb and not nozero;
236  5             mbuff(la) = 0;
237  5             la = la +1;
238  5             end;
239  4        end zero$mem;



240  3        DIAGNOSE: PROCEDURE;

241  4        DECLARE M BASED TA BYTE;

242  4        NEWLINE: PROCEDURE;
243  5           CALL CRLF; CALL PRINTADDR(TA); CALL PRINTCHAR(':');
246  5           CALL PRINTCHAR(' ');
247  5           END NEWLINE;

             /* PRINT DIAGNOSTIC INFORMATION AT THE CONSOLE */
248  4           CALL PRINT(.('LOAD  ADDRESS $')); CALL PRINTADDR(TA);
250  4           CALL PRINT(.('ERROR ADDRESS $')); CALL PRINTADDR(LA);

252  4           CALL PRINT(.('BYTES READ:$')); CALL NEWLINE;
254  4           DO WHILE TA < LA;
255  5           IF (LOW(TA) AND 0FH) = 0  THEN CALL NEWLINE;
257  5           CALL PRINTHEX(MBUFF(TA)); TA=TA+1;
259  5           CALL PRINTCHAR(' ');
260  5           END;
261  4        CALL CRLF;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
262  4          CALL BOOT;
263  4          END DIAGNOSE;
264  3       write$record: procedure;

265  4          call setdma(write$add);
266  4          if diskwrite(.fcba) <> 0 then call diskerror;
268  4          p = p+1;
269  4          end write$record;



270  3       empty$buffers:  procedure;
271  4            write$add = .memory;
272  4            do while write$add+127 <= (.memory+fa);
273  5               call write$record;
274  5               write$add = write$add+128;
275  5               end;
276  4            if not multi$segments then
277  4            do;
278  5               call write$record;
279  5               return;
280  5            end;
281  4            call move (write$add,.memory,(la:=.memory+fa+1-write$add));
282  4            end empty$buffers;



283  3            READHEX: PROCEDURE BYTE;
                     /* READ ONE HEX CHARACTER FROM THE INPUT */
284  4               declare khex byte;
285  4            if  (khex := check$one$hex(getchar)) <> 0ffh then return khex;
                   else
287  4                    DO; CALL PRINT(.('INVALID HEX DIGIT$'));
289  5                    CALL DIAGNOSE;
290  5                   end;
291  4            end readhex;

292  3            READBYTE: PROCEDURE BYTE;
                     /* READ TWO HEX DIGITS */
293  4               RETURN SHL(READHEX,4) OR READHEX;
294  4               END READBYTE;

295  3            READCS: PROCEDURE BYTE;
                     /* READ BYTE WHILE COMPUTING CHECKSUM */
296  4               DECLARE B BYTE;
297  4               CS = CS + (B := READBYTE);
298  4               RETURN B;
299  4               END READCS;



300  3       hex$input: procedure;
301  4            if rt = 2 or rt > 84h then
302  4            segadjst = shl(make$double(readcs,readcs),4); else

303  4               do;
                                 /* PROCESS EACH BYTE */
304  5                  DO WHILE (RL := RL - 1) <> 255;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
305  6                CALL SETMEM(READCS); LA = LA+1;
307  6                END;
308  5             IF LA > FA THEN FA = LA - 1;
310  5             end;



                           /* NOW READ CHECKSUM AND COMPARE */
311  4                     IF CS + READBYTE <> 0 THEN
312  4                     DO; CALL PRINT(.('CHECK SUM ERROR $'));
314  5                        CALL DIAGNOSE;
315  5                        END;
316  4             end hex$input;


317  3     get$buffer$len: procedure;
318  4          multi$segments = true;
319  4          if rt = 84h then rt = 83h;
321  4          else if rt = 83h then rt = 84h;
               if seg$length (rt-81h) <= (high$add:=la+rl-1) then
324  4             do;
325  5             if high$add=0 then high$add = 1;
327  5             seg$length (rt-81h) = high$add;
328  5             header(rt-81h).typseg = rt-80h;
329  5             end;
330  4          end get$buffer$len;



               /* INITIALIZE */
331  3          SA, FA, NB = 0;
332  3          P = 0; /* PARAGRAPH COUNT */
333  3          SBUFF(0) = EOFILE;
334  3          fcb(nr) = 0;
335  3          if head then fcb(nr) = 1;
337  3          multi$segments = false;
338  3          segadjst = 0;
339  3          do k= 0 to 7;
340  4             seglength(k) = 0;
341  4             end;

342  3          call zero$mem;

343  3          ta=0;
344  3          la=1;
               /* READ RECORDS UNTIL :00XXXX IS ENCOUNTERED */

345  3             DO FOREVER;
                  /* SCAN THE : */
346  4                DO WHILE (nxb:=getchar) <> ':';
347  5                  if nxb = eofile then go to second;
                        /* MAY BE THE END OF TAPE */
349  5                END;

                  /* SET CHECK SUM TO ZERO, AND SAVE THE RECORD LENGTH */
350  4                CS = 0;
351  4                      nb = nb +(rl:=readcs);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
352  4                  TA, LA = MAKE$DOUBLE(READCS,READCS) + segadjst;
353  4                  IF SA = 0 THEN SA = LA;


                        /* READ THE RECORD TYPE  */

                        /*   skip all records except type 0 2 31  */
355  4                  if (rt:=readcs) > digital$code and rt < digital02 then
356  4                     do;
357  5                     if not t8080 then
358  5                       call get$buffer$len;  else
359  5                       call hex$input;
360  5                     end;    else
361  4                        do;
362  5                        if (rt = digital$code) then
363  5                            do;
364  6                            call hex$input;
365  6                            header(0).typseg = 1;
366  6                            end;   else
367  5                               do;
368  6                                 if (rt = 0 and la < segmts(1).begin$add and la >= segmts(0).begin$add)
                                       or rt = 2 then
369  6                                     do;
370  7                                     la = la-segmts(0).begin$add;
371  7                                     call hex$input;
372  7                                     header(0).typseg = 1;
373  7                                     end;
374  6                                  if (rt = 0 and la >= segmts(1).begin$add) then
375  6                                       do;
376  7                                       multi$segments = true;
377  7                                       if seg$length(1) <
                                             (high$add:=la+rl-segmts(1).begin$add-1) then
378  7                                          do;
379  8                                          seg$length(1) = high$add;
380  8                                          header(1).typseg=2;
381  8                                          end;
382  7                                       end;
383  6                               end;
384  5                        end;
385  4              end;


386  3     second:
                  call empty$buffers;
387  3                 ta = (la+paragraph-1) and 0fff0h;
388  3                 header(0).file$length=fa/16+1;
389  3                 if header(0).minimum$mem = 0 then header(0).minimum$mem = fa/16+1;
391  3                 fa=ta;
392  3                 if not multi$segments then go to fin;
394  3                 call zero$mem;
395  3                 multi$segments = false;
396  3                 sfcb(ex),sfcb(nr) = 0;
397  3                 call open(.sfcb);
398  3                 call setdma(.buffer);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
397  3          do K = 1 to 7;
400  4             if seg$length(K) <> 0 then
401  4             do;
402  5               seg$length(K) = seg$length(K)+paragraph and 0fff0h;
403  5               header(K).file$length = seg$length(K)/16;
404  5               if header(K).minimum$mem=0 then
405  5               header(K).minimum$mem=seg$length(K)/16;
406  5               end;
407  4             end;
408  3          segadjst = 0;
409  3          seg$length(0) = ta;
410  3          sbp=length(sbuff);


411  3          DO FOREVER;
                /* SCAN THE : */
412  4              DO WHILE (nxb:=getchar) <> ':';
413  5                if nxb = eofile then go to afin;
415  5              END;

416  4          cs = 0;
417  4          rl = readcs;

418  4          la = segadjst+make$double(readcs,readcs);

419  4          if (rt := readcs) = eofile then go to afin;
421  4        if rt = 84h then rt = 83h;
423  4        else if rt = 83h then rt = 84h;
                if rt > digital$code and rt < digital02 then
426  4             do;
427  5               do K = 0 to (rt-82h);
428  6               la = la + seg$length(K);
429  6               end;
430  5             call hex$input;
431  5             end;
432  4             if (rt = 0 and la > segmts(1).begin$add) or rt = 2 then
433  4             do;
434  5               la = la - segmts(1).begin$add + seg$length(0);
435  5               call hex$input;
436  5               end;


437  4          END;


438  3    afin:
             call empty$buffers;


439  3        FIN:
             /* PRINT FINAL STATISTICS */
             CALL PRINT(.('BYTES READ    $')); CALL PRINTADDR(NB);
441  3        CALL PRINT(.('RECORDS WRITTEN $')); CALL PRINTHEX(P+1);
443  3        CALL CRLF;

             /*   write the header record   */
444  3        call close(.fcba);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
445  3            if head then
446  3            do;
447  4               fcb(ex),fcb(nr) = 0;
448  4               call open(.fcba);
449  4               call move (.header,.buffer,128);
450  4               call setdma(.buffer);
451  4               if diskwrite(.fcba) <> 0 then call diskerror;

453  4            end;
454  3         END RELOC;


455  2      declare seg$number byte;

456  2      ignore$filename: procedure;
457  3           tbp = 0;
458  3           char = buffer(tbp);
459  3           call next$non$blank;
460  3           do while (char:=buffer(tbp)) <> ' ';
461  4           tbp = tbp +1;
462  4           end;

463  3      end ignore$filename;



464  2      parse$tail: procedure;
465  3         declare seg$index byte;

466  3         get$segmt: procedure byte;
               /*   get the segment name   */
467  4            declare ( Kentry, match$flag,j, no$match) byte;
468  4            declare user$segmt(5)  byte;

469  4            do j = 0 to last (user$segmt);
470  5               if delimiter then
471  5               user$segmt(j) = ' ';  else
472  5                 do;
473  6                 user$segmt(j) = char;
474  6                 char = tran(retchar);
475  6                 end;
476  5               end;


477  4            seg$index = 0;
478  4            no$match, matchflag = true;

479  4            do while no$match and seg$index < 11;

480  5                match$flag=true;
481  5                Kentry = 0;
482  5                do while match$flag and Kentry <= last (segmts.name);
483  6                   if usersegmt(Kentry) <> segmts(seg$index).name(Kentry) then
484  6                      matchflag = false;  else
485  6                      Kentry = Kentry +1;
486  6                      end;
487  5                if matchflag then no$match = false; else
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
487  5                seg$index = seg$index +1;
490  5              end;
491  4              if no$match then seg$index = 0ffh;
493  4              return seg$index;
494  4          end get$segmt;

495  3      get$switches: procedure;
496  4        do while char <> ']' and char <> cr;
497  5           call next$non$blank;
498  5          if char= 'A' then header(seg$index).absolute$add = (get$num);
500  5          else if
                char= 'M' then
501  5              do;
502  6              header(seg$index).minimum$mem = (get$num);
503  6              header(seg$index).typseg = seg$index+1;
504  6              end;
505  5          else if
                char= 'X' then header(seg$index).maximum$mem = (get$num);
507  5          else if
                char= 'B' then segmts(seg$index).begin$add = (get$num$16);
509  5          else do;
510  6             call comline$error;
511  6             call boot;
512  6           end ;
513  5         end;


514  4      end get$switches;

515  3        do forever;
516  4           call next$non$blank;
517  4           if char = cr then return;
519  4           if  get$segmt = 0ffh then
520  4           do;
521  5             call comline$error;
522  5           end;
523  4           if seg$index < 8 then
524  4           do;
525  5              if char = ']' or char = cr then call comline$error;
527  5              call get$switches;
528  5           end;
                else
529  4           do;
530  5              if seg$index = 8 then t8080 = true; else
532  5              do;
533  6                 if seg$index = 9 then nozero = true; else
535  6                  head = false;
536  6              end;
537  5           end;
538  4         end;

539  3        end parse$tail;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                /* ARRIVE HERE FROM THE SYSTEM MONITOR, READY TO READ THE HEX TAPE */

                /* SET UP STACKPOINTER IN THE LOCAL AREA */
540   2         DECLARE STACK(64) ADDRESS;
541   2         SP = STACKPOINTER; STACKPOINTER = .STACK(LENGTH(STACK));
543   2         LA = 0h;
544   2         mbuffadr = .memory;
545   2         t8080 = false;
546   2         nozero = false;
547   2         head = true;

548   2         SBP = LENGTH(SBUFF);
                     /* SET UP THE SOURCE FILE */
549   2            CALL MOVE(.FCBA,.SFCB,33);
550   2            CALL MOVE(.('M86' ),.SFCB(9),4);
551   2            CALL OPEN(.SFCB);
552   2            IF DCNT = 255 THEN CALL PERROR(.('CANNOT OPEN SOURCE$'));

554   2            CALL MOVE(.('CMD'),.FCBA+9,3);

                /* REMOVE ANY EXISTING FILE BY THIS NAME */
555   2         CALL DELETE(.FCBA);
                /* THEN OPEN A NEW FILE */
556   2         CALL MAKE(.FCBA); CALL OPEN(.FCBA);
558   2         IF DCNT = 255 THEN CALL PERROR(.('NO MORE DIRECTORY SPACE$')); ELSE
560   2                DO;
561   3                call ignore$filename;
562   3                call parse$tail;
563   3                CALL RELOC;
564   3                CALL CLOSE(.FCBA);
565   3                IF DCNT = 255 THEN CALL PERROR(.('CANNOT CLOSE FILE$'));
567   3                END;
568   2         CALL CRLF;

569   2         CALL BOOT;
570   2        END plm;
571   1        END;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES										
7	0000H	2	A.	WORD PARAMETE	8								
64	0004H	2	A.	WORD PARAMETER AUTOMATIC		65	66	67					
69	0004H	2	A.	WORD PARAMETER AUTOMATIC		70	71						
4	0000H	2	A.	WORD PARAMETER	5								
80	0004H	2	A.	WORD PARAMETER AUTOMATIC		81	83						
73	0004H	2	A.	WORD PARAMETER AUTOMATIC		74	76						
132	0000H	1	A.	BYTE BASED(S)	134								
92	0004H	2	A.	WORD PARAMETER AUTOMATIC		93	94						
15	0003H	2	ABSOLUTEADD. . . .	WORD MEMBER(HEADER)	499								
438	0736H		AFIN	LABEL	414 420								
225	0004H	1	B.	BYTE PARAMETER AUTOMATIC		226	232						
296	05B3H	1	B.	BYTE	297 298								
132	0000H	1	B.	BYTE BASED(D)	134								
155	0004H	1	B.	BYTE PARAMETER AUTOMATIC		156	157	159	160	161			
59	0004H	1	B.	BYTE PARAMETER AUTOMATIC		60	61	62					
14	0005H	2	BEGINADD	WORD MEMBER(SEGMTS)	368	370	374	377	432	434	508		
11	0000H	14	BOOT	PROCEDURE STACK=0008H		86	148	230	262	511	569		
19			BSIZE.	LITERALLY	19								
2	0000H	128	BUFFA.	BYTE ARRAY(128) EXTERNAL(2)			18	21					
18	0000H	128	BUFFER	BYTE ARRAY(128) EXTERNAL(2) AT 460			144	152	208	398	449	450	458
139	05A7H	1	CHAR	BYTE	164 165 166 185 193 195 197 198 458 460 473 474 496 498 500 505 507 517 525								
45	0004H	1	CHAR	BYTE PARAMETER AUTOMATIC		46	47						
169	0323H	36	CHECKONEMEX. . . .	PROCEDURE BYTE STACK=0004H			195	285					
100	01E9H	19	CLOSE.	PROCEDURE STACK=000AH		444	564						
140	0299H	51	COMLINEERROR . . .	PROCEDURE STACK=0012H		196	510	521	526				
3	0000H	38	COPYRIGHT.	BYTE ARRAY(38) DATA									
21	0000H	1	COUNTCOMMANDTAIL .	BYTE EXTERNAL(2) AT	151								
23	—		CR	LITERALLY	50 496 517 525								
49	0113H	17	CRLF	PROCEDURE STACK=000EH		75	142	147	243	261	443	568	
221	05ACH	1	CS	BYTE	297 311 350 416								
131	0006H	2	D.	WORD PARAMETER AUTOMATIC		132	134	136					
18			DBUFF.	LITERALLY									
91	05A3H	1	DCNT	BYTE	98 102 106 109 125 552 558 565								
183	0026H	5	DEL.	BYTE ARRAY(5) DATA	184 185								
111	0221H	16	DELETE	PROCEDURE STACK=000AH		555							
181	035EH	44	DELIMITER.	PROCEDURE BYTE STACK=0002H			194	470					
17			DFCBA.	LITERALLY									
240	07F9H	90	DIAGNOSE	PROCEDURE STACK=0024H		289	314						
2			DIGITAL02.	LITERALLY	355 425								
2			DIGITALCODE. . . .	LITERALLY	355 362 425								
2			DIGITALDATA. . . .	LITERALLY									
88	01BAH	12	DISKERROR.	PROCEDURE STACK=0026H		267	452						
115	0231H	16	DISKREAD	PROCEDURE BYTE STACK=000AH			207						
119	0241H	16	DISKWRITE.	PROCEDURE BYTE STACK=000AH			266	451					
			DOUBLE	BUILTIN	179								
270	088CH	81	EMPTYBUFFERS . . .	PROCEDURE STACK=002EH		386	438						
19			EOFILE	LITERALLY	212 333 347	413	419						

COPYRIGHT 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

```
  2                EX . . . . . LITERALLY      396  447
  4  0000H    1    F. . . . . BYTE PARAMETER        5
  7  0000H    1    F. . . . . BYTE PARAMETER        8
223  0012H    2    FA . . . . . WORD       272  281  308  309  331  388  390  391
 23                FALSE. . . . . LITERALLY      188  337  395  484  488  535  545  546
119  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC       120  121
100  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC       101  102
104  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC       105  106
123  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC       124  125
127  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC       128  129
115  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC       116  117
111  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC       112  113
 17  0000H   33    FCB. . . . . BYTE ARRAY(33) EXTERNAL(1) AT      334  336  447
 96  0004H    2    FCB. . . . . WORD PARAMETER AUTOMATIC        97   98
  2  0000H   33    FCBA . . . . BYTE ARRAY(33) EXTERNAL(1)          17  266  444  448  451  549  554
                                 555  556  557  564
 15  0001H    2    FILELENGTH . . . . WORD MEMBER(HEADER)       388  403
439  0739H        FIN. . . . . LABEL      393
 23                FOREVER. . . . . LITERALLY      345  411  515
  1  0000H        GENCMD . . . . . PROCEDURE STACK=0000H
317  0784H  114    GETBUFFERLEN . . . PROCEDURE STACK=0004H       358
202  03CFH  124    GETCHAR. . . . . PROCEDURE BYTE STACK=0026H        285  346  412
190  033AH   69    GETNUM . . . . . PROCEDURE WORD STACK=0016H        499  502  506  508
466  0A7EH  201    GETSEGMT . . . . . PROCEDURE BYTE STACK=0008H      519
495  0847H  180    GETSWITCHES. . . . PROCEDURE STACK=001AH       527
177  0006H    1    H. . . . . BYTE PARAMETER AUTOMATIC         178  179
169  0004H    1    H. . . . . BYTE PARAMETER AUTOMATIC         170  171  172  173  175
 79  05A5H    1    HEAD . . . . . BYTE       335  445  535  547
 15  00F9H  135    HEADER . . . . . STRUCTURE ARRAY(15) INITIAL      328  365  372  380  388  389  390
                                 403  404  405  449  499  502  503  506
300  0924H   96    HEXINPUT . . . . . PROCEDURE STACK=003AH       359  364  371  430  435
                   HIGH . . . . . BUILTIN          66
224  002AH    2    HIGHADD. . . . . WORD       323  325  326  327  377  379
 25  05A4H    1    I. . . . . BYTE        26   27   28   29   30   31   32   33   34   35   36   37
                                  38   39   40   41   42   43
203  05AAH    1    I. . . . . BYTE       207  210
182  05A9H    1    I. . . . . BYTE       184  185
141  05A8H    1    I. . . . . BYTE       143  144
456  09F6H   45    IGNOREFILENAME . . PROCEDURE STACK=000CH       561
467  05B8H    1    J. . . . . BYTE       469  471  473
221  05AEH    1    K. . . . . BYTE       339  340  399  400  402  403  404  405  427  428
467  05B4H    1    KENTRY . . . . . BYTE       481  482  483  485
284  05B2H    1    KHEX . . . . . BYTE       285  286
177  0004H    1    L. . . . . BYTE PARAMETER AUTOMATIC         178  179
 78  0008H    2    LA . . . . . WORD        85  227  232  235  236  237  251  254  281  306  308  309
                                 323  344  352  354  368  370  374  377  387  418  428  432  434  543
                   LAST . . . . . BUILTIN          184  204  206  213  469  482
                   LENGTH . . . . . BUILTIN          410  542  548
 23                LF . . . . . LITERALLY       51
                   LOW. . . . . BUILTIN          67  255
241  0000H    1    M. . . . . BYTE BASED(TA)
123  0251H   19    MAKE . . . . . PROCEDURE STACK=000AH       556
177  0347H   23    MAKEDOUBLE . . . . PROCEDURE WORD STACK=0006H          302  352  418
467  05B7H    1    MATCHFLAG. . . . . BYTE       478  480  482  484  487
  2  0000H    2    MAXB . . . . . WORD EXTERNAL(0)          227  235
 15  0007H    2    MAXIMUMMEM . . . . WORD MEMBER(HEADER)       506
223  0000H  256    MBUFF. . . . . BYTE BASED(MBUFFADR) ARRAY(256)          232  236  257
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 78  0006H    2  MBUFFADR . . . . .  WORD       223  232  236  257  544
     0000H       MEMORY . . . . . .  BYTE ARRAY(0)       227  235  271  272  281  544
 15  0005H    2  MINIMUMMEM . . . .  WORD MEMBER(HEADER)       389  390  404  405  502
  4  0000H       MON1 . . . . . . .  PROCEDURE EXTERNAL(3) STACK=0000H        12   47   71   94  113  129
  7  0000H       MON2 . . . . . . .  PROCEDURE BYTE EXTERNAL(4) STACK=0000H        98  102  106  109  117
                                      121  125
131  0274H   37  MOVE . . . . . . .  PROCEDURE STACK=0008H          208  281  449  549  550  554
222  05A5H    1  MULTISEGMENTS. . .  BYTE       276  318  337  376  392  395
 53  0004H    1  N. . . . . . . . .  BYTE PARAMETER AUTOMATIC        54   55   56   57
131  0004H    1  N. . . . . . . . .  BYTE PARAMETER AUTOMATIC       132  133
 14  0000H    5  NAME . . . . . . .  BYTE ARRAY(5) MEMBER(SEGMTS)        482  483
223  0014H    2  NB . . . . . . . .  WORD       331  351  440
242  0853H   27  NEWLINE. . . . . .  PROCEDURE STACK=0020H          253  256
163  030DH   22  NEXTNONBLANK . . .  PROCEDURE STACK=0008H          459  497  516
467  05B9H    1  NOMATCH. . . . . .  BYTE       478  479  488  491
 22  05A3H    1  NOZERO . . . . . .  BYTE       235  534  546
  2              NR . . . . . . . .  LITERALLY       334  336  396  447
223  05B0H    1  NXB. . . . . . . .  BYTE       346  347  412  413
 96  01D6H   19  OPEN . . . . . . .  PROCEDURE STACK=000AH          397  448  551  557
223  05B1H    1  P. . . . . . . . .  BYTE       268  332  442
191  000AH    2  PARADD . . . . . .  WORD       192  197  200
  2              PARAGRAPH. . . . .  LITERALLY       387  402
464  0A23H   91  PARSETAIL. . . . .  PROCEDURE STACK=001EH          562
 24  00C2H   62  PATCH. . . . . . .  PROCEDURE STACK=0002H
 80  0195H   37  PERROR . . . . . .  PROCEDURE STACK=0022H          89  211  553  559  566
 16  000EH  180  PLM. . . . . . . .  PROCEDURE PUBLIC STACK=0042H
 73  0185H   16  PRINT. . . . . . .  PROCEDURE STACK=0014H           82  229  248  250  252  288  313  439
                                      441
 64  015EH   23  PRINTADDR. . . . .  PROCEDURE STACK=001CH           85  244  249  251  440
 45  0100H   19  PRINTCHAR. . . . .  PROCEDURE STACK=000AH           50   51   56   57  144  146  245  246
                                      259
 37  0143H   27  PRINTHEX . . . . .  PROCEDURE STACK=0016H           66   67  257  442
 69  0175H   16  PRINTM . . . . . .  PROCEDURE STACK=000AH           76   83   84
 53  0124H   31  PRINTNIB . . . . .  PROCEDURE STACK=0010H           61   62
292  08FFH   19  READBYTE . . . . .  PROCEDURE BYTE STACK=0030H        297  311
295  0912H   18  READCS . . . . . .  PROCEDURE BYTE STACK=0034H        302  305  351  352  355  417  418
                                      419
283  08DDH   34  READHEX. . . . . .  PROCEDURE BYTE STACK=002AH        293
220  044BH  851  RELOC. . . . . . .  PROCEDURE STACK=003EH          563
127  0264H   16  RENAME . . . . . .  PROCEDURE STACK=000AH
150  02CCH   32  RETCHAR. . . . . .  PROCEDURE BYTE STACK=0002H        164  166  193  198  474
 19  05A1H    1  RFLAG. . . . . . .  BYTE
221  05ABH    1  RL . . . . . . . .  BYTE       304  323  351  377  417
221  05A0H    1  RT . . . . . . . .  BYTE       301  319  320  321  322  323  327  328  355  362  368  374
                                      419  421  422  423  424  425  427  432
131  0008H    2  S. . . . . . . . .  WORD PARAMETER AUTOMATIC       132  134  135
223  0010H    2  SA . . . . . . . .  WORD       331  353  354
 19  0002H    2  SBP. . . . . . . .  WORD       204  205  206  208  212  213  216  410  548
 19  01A1H 1024  SBUFF. . . . . . .  BYTE ARRAY(1024)        204  205  206  208  212  213  217  333  410
                                      548
101  01FCH   19  SEARCH . . . . . .  PROCEDURE STACK=000AH
108  020FH   18  SEARCHN. . . . . .  PROCEDURE STACK=0008H
386  05ADH       SECOND . . . . . .  LABEL      348
223  0016H    2  SEGADJST . . . . .  WORD       302  338  352  408  418
465  0585H    1  SEGINDEX . . . . .  BYTE       477  479  483  489  492  493  499  502  503  506  508  523
                                      530  533
223  0018H   16  SEGLENGTH. . . . .  WORD ARRAY(8)       323  327  340  377  379  400  402  403  405  409
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                                428  434
   14  00ACH    77  SEGMTS . . . . . .  STRUCTURE ARRAY(11) INITIAL      368  370  374  377  432  434  482
                                                483  508
  455  05B4H     1  SEGNUMBER. . . . .  BYTE
   92  01C6H    16  SETDMA . . . . . .  PROCEDURE STACK=000AH       265  398  450
  225  079EH    42  SETMEM . . . . . .  PROCEDURE STACK=001AH       305
   19  0180H    33  SFCB . . . . . .  BYTE ARRAY(33)      207  396  397  549  550  551
                    SHL. . . . . .  BUILTIN        179  293  302
                    SHR. . . . . .  BUILTIN         61
   10  0000H     2  SP . . . . . .  WORD       541
  540  002CH   128  STACK. . . . . .  WORD ARRAY(64)      542
  219               STACKPOINTER . . .  LITERALLY      541
                    STACKPTR . . . . .  BUILTIN        541  542
   22  05A2H     1  T8080. . . . . .  BYTE       357  531  545
  223  000EH     2  TA . . . . . .  WORD       241  244  249  254  255  257  258  343  352  387  391  409
  223  000CH     2  TABS . . . . . .  WORD
   20  0004H     2  TBP. . . . . .  WORD       143  151  152  457  458  460  461
  155  02ECH    33  TRAN . . . . . .  PROCEDURE BYTE STACK=0004H       164  166  474
   23               TRUE . . . . . .  LITERALLY      186  318  345  376  411  478  480  515  531  534  547
   15  0000H     1  TYPSEG . . . . . .  BYTE MEMBER(HEADER)      328  365  372  380  503
  468  05BAH     5  USERSEGNT. . . . .  BYTE ARRAY(5)       469  471  473  483
   23               WHAT . . . . . .  LITERALLY
  223  0028H     2  WRITEADD . . . . .  WORD       265  271  272  274  281
  264  086EH    30  WRITERECORD. . . .  PROCEDURE STACK=002AH       273  278
  234  07C8H    49  ZEROMEM. . . . . .  PROCEDURE STACK=0002H       342  394
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 0BFBH   3067D
    CONSTANT AREA SIZE = 0131H    305D
    VARIABLE AREA SIZE = 05BFH   1471D
    MAXIMUM STACK SIZE = 0042H     66D
    773 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____